4

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 02-146630
(43)Date of publication of application : 05.06.1990

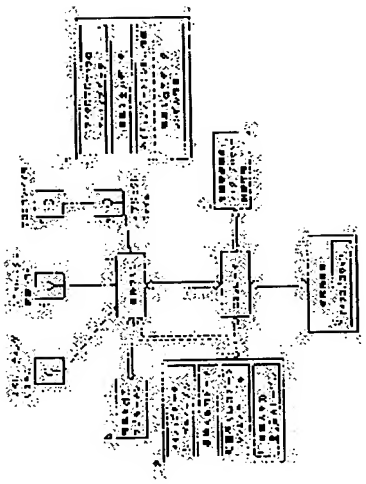
(51)Int.Cl. G06F 9/455
G06F 11/22

(21)Application number : 63-299668
(22)Date of filing : 29.11.1988

(71)Applicant : FUJITSU LTD
FUJITSU MICROCOMPUTER SYST LTD
(72)Inventor : OBA SHIRO

(54) PROGRAM DEVELOPING SYSTEM FOR MICROPROCESSOR

(57)Abstract:
PURPOSE: To attain a debugging job peculiar to a high-level language even in a cross development environment where an emulator is used by reading the execution information via the emulator at every instruction and executing the corresponding standard input/output subroutine via a developing machine when the coincidence is obtained between the execution information and the information on an interruption table.
CONSTITUTION: The program of a microprocessor 13 of a device 3 to be developed is executed by an emulator 2 based on the native control information on the microprocessor 13. At the same time, this execution information is compared with the standard input/output information 9 via the emulator 2. When the coincidence is obtained between both bits of information, the emulator 2 stops the execution of the program of the microprocessor 13. At the same time, the information on the standard input/output is sent to a developing machine 1 to substitute the execution of a standard input/output function of the machine 1. Thus it is possible to carry out a debugging job in the program of the microprocessor 13 while viewing the CRT of a personal computer similarly to a self-development environment even in a cross development environment.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平2−146630

⑤Int.Cl.³	識別記号	庁内整理番号	㊸公開 平成2年(1990)6月5日
G 06 F 9/455 11/22	340 A	7368−5B 8724−5B	G 06 F 9/44 310 A

審査請求 未請求 請求項の数 1 (全13頁)

㊹発明の名称　マイクロプロセッサのプログラム開発システム

㉑特　願　昭63−299668
㉒出　願　昭63(1988)11月29日

㉒発明者　大庭　史朗　神奈川県川崎市中原区上小田中1015番地　富士通マイコンシステムズ株式会社内
㉑出願人　富士通株式会社　神奈川県川崎市中原区上小田中1015番地
㉑出願人　富士通マイコンシステムズ株式会社　神奈川県川崎市中原区上小田中1015番地
㉔代理人　弁理士　青木　朗　外4名

明　細　書

1. 発明の名称

マイクロプロセッサのプログラム開発システム

2. 特許請求の範囲

少なくともオペレーティングシステムとエミュレータを制御するシステムを内蔵し，被開発装置に於けるマイクロプロセッサのプログラムの開発を実行する開発マシン，該開発マシンにより制御されるエミュレータ及び該エミュレータに接続され，該エミュレータによりそのプログラムが制御されるマイロプロセッサを有する被開発装置とから構成され，且つ該開発マシンのマイクロプロセッサと該被開発装置に於けるマイクロプロセッサとが異なる機械語で動作するクロス開発環境下での被開発装置におけるマイクロプロセッサのプログラム開発システムにおいて，該開発マシンにロードされるコンパイラが出力するオブジェクトファイル内に予め該被開発装置のマイロプロセッサ制御用情報及び標準入出力機能を実行させる標準入出力情報とを含ませておく手段，該オブジェクトファイルが当該システムにロードされた時に該被開発装置のマイロプロセッサの制御情報をエミュレータのメモリにストアしておく手段，該標準入出力機能を実行するためサブルーチンを含む標準入出力情報を該開発マシンのメモリにストアする手段，及び該標準入出力情報に関する標準入出力シンボルテーブルと標準入出力割り込みテーブルとを該開発マシンのメモリか該エミュレータのメモリの何れかにストアする手段とをそれぞれ設けるとともに，該被開発装置のマイロプロセッサのプログラムの実行情報と該標準入出力割り込みテーブルの情報とを比較検出する手段を該エミュレータに設けておき，該エミュレータが該被開発装置のマイロプロセッサのプログラムを制御し実行させるに際し1命令毎に当該プログラムの実行情報をエミュレータにて読みとり，該比較検出手段において当該実行情報と該割り込みテーブルの情報とを比較し，両者が一致した場合にはエミュレータは該被開発装置のマイロプロセッサのプロ

グラムの実行停止を指示すると共に，当該標準入出力に関する関数情報を該標準入出力シンボルテーブルから選択して該開発マシンへ送り，該標準入出力関数情報に該当する標準入出力サブルーチンを該開発マシンに実行させることを特徴とする被開発装置のマイクロプロセッサのプログラム開発システム

3. 発明の詳細な説明

〔概　要〕

被開発装置に於けるマイクロプロセッサのプログラム開発システムに関し、

クロス開発環境においてもセルフ開発環境において一般的に行われている標準入出力関数を用いてのプログラム評価を被開発装置におけるマイクロプロセッサのシステム評価にても行えるようにすることを目的とし、少なくともオペレーティングシステムとエミュレータを制御するシステムを内蔵し，被開発装置に於けるマイクロプロセッサのプログラムの開発を実行する開発マシン，該開発マシンにより制御されるエミュレータ及び該エミュレータに接続され，該エミュレータによりそのプログラムが制御されるマイロプロセッサを有する被開発装置とから構成されたクロス開発環境下での被開発装置のマイロプロセッサのプログラム開発において，該開発マシンにロードされるコンパイラが出力するオブジェクトファイル内に予め該被開発装置のマイロプロセッサ制御用情報及び標準入出力機能を実行させる標準入出力情報とを含ませておく手段，該オブジェクトファイルが当該システムにロードされた時に該被開発装置のマイロプロセッサの制御情報をエミュレータのメモリにストアしておく手段，該標準入出力機能を実行するためサブルーチンを含む標準入出力情報を該開発マシンのメモリにストアする手段，及び該標準入出力情報に関する標準入出力シンボルテーブルと標準入出力割り込みテーブルとを該開発マシンのメモリか該エミュレータのメモリの何れかにストアする手段とをそれぞれ設けるとともに，該被開発装置のマイロプロセッサのプログラムの実行情報と該標準入出力割り込みテーブルの情報

（3）

（4）

とを比較検出する手段を該エミュレータに設けておき，該エミュレータが該被開発装置のマイロプロセッサのプログラムを制御し実行させるに際し1命令毎に当該プログラムの実行情報をエミュレータにて読みとり，該比較検出手段において当該実行情報と該割り込みテーブルの情報とを比較し，両者が一致した場合にはエミュレータは該被開発装置のマイロプロセッサのプログラムの実行停止を指示すると共に，当該標準入出力に関する関数情報を該標準入出力シンボルテーブルから選択して該開発マシンへ送り，該標準入出力関数情報に該当する標準入出力サブルーチンを該開発マシンに実行させるように構成する。

〔産業上の利用分野〕

本発明はマイクロコンピュータの制御ソフトウェアを開発する際の例えば高級言語を用いたクロスコンパイラの利用をより容易にするためのマンマシンインターフェースを向上するためのもので、マイクロコンピュータの制御プログラム開発の分野で利用されるものである。

又より具体的には、被開発装置に於けるマイクロプロセッサのプログラムの開発を実行する為にマイクロプロセッサを内蔵している開発マシン（以下開発マシンと言う）を使用して，該開発マシンのマイクロプロセッサと被開発装置に於けるマイクロプロセッサとが互換性のない異なる機械語で動作するクロス開発環境下（以下単にクロス開発環境下と言う）において被開発装置のマイロプロセッサ等のマイクロコンピュータのプログラムを開発及び評価する場合においても高級言語のようなクロスコンパイラを用いてセルフ開発環境下におけると同様の手法でその開発評価が出来るシステムに関するものである。

〔従来の技術〕

マイクロコンピュータを内蔵した被開発装置，例えば温風ヒータとかエアコン等はこれ等の製品が出荷される以前に、組み込まれたマイロプロセッサのコンピュータプログラムが正常に作動する

（5）

（6）

かどうかを検査し評価することが一般的に行われている。

従来、このようなマイクロコンピュータの制御ソフトウェア開発環境はアセンブラ一辺倒であったが、近年高級言語、それもC言語を使用するプログラマが増加し、それに伴い高級言語デバッグをサポートするソフトウェアシュミレータが出現している。

処で従来においてこの様な被開発装置のマイロプロセッサのプログラムを開発するに際しては通常セルフ開発環境の下で実行されている。セルフ開発環境とは被開発装置のマイロプロセッサと開発マシン例えばパソコンのマイクロプロセッサとが同一種類の機械語で動作する場合を云い、一般的にはパソコンのマイクロプロセッサが被開発装置のマイロプロセッサを制御しながら評価が行われるものであり、アセンブラにより機械語と1対1に対応して実行されていた。然しながら実際の作業としては、マイロプロセッサにおけるプログラムが正しく動作しているかをみるには該マイク

ロプロセッサの動作をいちいちブレークしてそのたびに調べたい値についての変数をメモリをのぞいてその値をみる必要がある。かかるやり方はかなり辛抱強さがいる。つまりこのシステムでは変数変化が予想されるアドレスを予め指定してそこで実行を停めて別のコマンドを用いてメモリの中を見ることにより確認しそれから別の実行に移るという繁雑さが要求されていた。

かかる繁雑さを解消する方法としてコンパイラつまり高級言語例えばC言語を用いてデバッグすることが盛んになりアセンブラを知らなくても被開発装置に於けるマイクロプロセッサのプログラム開発が出来るようになって来た。そして更に、パソコン側のマイクロプロセッサに被開発装置のマイロプロセッサの制御を代行させ，例えばC言語のコンパイラで上記の開発を行う場合には、プログラムの実行において高級言語が正しいか否かを調べるのにプログラムの途中に標準入出力関数例えば"printf"文を挿入しておき、その"printf"文により要求された変数をパソコンのCRT画面

に表示させて確認するというような方法で実行されており、手軽な方法としてよく使用されている。この方法ではセルフ開発であるため動作する被開発装置のマイロプロセッサと開発マシンのマイクロプロセッサとが同一種類の機械語で動作するものであり、従って該被開発装置のマイロプロセッサの動作をパソコンのマイクロプロセッサで代行することが可能となるのである。

又このような高級言語を扱っている人はいちいちプログラムの実行を停めずに調べたい変動データや情報をCRTに表示して確認する方法に慣れているのが一般的である。

一方かかる評価システムにおいてセルフ開発環境とは異なりデバッグするプログラムを調べるパソコン用のマイクロプロセッサと該被開発装置のマイロプロセッサとが異なる種類の機械語で動作する場合のプログラム開発を行う即ちクロス開発環境の場合には、パソコンのマイクロプロセッサで代行する事が出来ない。

従って，前述したように被開発装置のシステム

の制御は直接には該被開発装置のマイロプロセッサが行い，パソコン側のマイクロプロセッサがこの被開発装置のマイロプロセッサをエミュレータを介して止めたり，走らせたりすることによって間接的に被開発装置のシステムの制御を行う方式を取らざるを得なくなる。

その場合において，プログラムの途中に例えば"printf"文を挿入して実行させたとしてもマイロプロセッサに通常表示機能がないため直ちにその結果を表示することは出来ない。

然しながら前述のように高級言語による評価に慣れている人の要望もあり、クロス開発環境下でもセルフ開発環境と同じような評価システムが確立されることが望まれていた。

〔発明が解決しようとする課題〕

従って本発明の目的はクロス開発環境下においてもセルフ開発環境において一般的に行われている標準入出力関数を用いて被開発装置のマイクロプロセッサのプログラム開発を行うことの出来る

方法を提供しようとするものである。

〔課題を解決するための手段〕

上記の目的を達成するため、現物のマイロプロセッサのシステムを評価するにはエミュレータを用いて評価する必要があり、高級言語をサポートしているクロスコンパイラ及びそれに対応したエミュレータを用意してプログラム開発や評価にセルフ開発環境と同一のデバッグ手法をとることを可能にする必要があり、かかる技術的課題を実現するために次のような技術的構成を採用したものである。即ち、

少なくともオペレーティングシステムとエミュレータを制御するシステムを内蔵し、被開発装置に於けるマイクロプロセッサのプログラムの開発を実行する開発マシン（以下単に開発マシンと言う）、該開発マシンにより制御されるエミュレータ及び該エミュレータに接続され、該エミュレータによりそのプログラムが制御されるマイロプロセッサを有する被開発装置とから構成され、且つ

該開発マシンのマイクロプロセッサと該被開発装置に於けるマイクロプロセッサとが異なる機械語で動作するクロス開発環境下（以下単にクロス開発環境下と言う）での被開発装置におけるマイクロプロセッサのプログラム開発システムにおいて、該開発マシンにロードされるコンパイラが出力するオブジェクトファイル内に予め該被開発装置のマイロプロセッサ制御用情報及び標準入出力機能を実行させる標準入出力情報とを含ませておく手段、該オブジェクトファイルが当該システムにロードされた時に該被開発装置のマイロプロセッサの制御情報をエミュレータのメモリにストアしておく手段、該標準入出力機能を実行するためサブルーチンを含む標準入出力情報を該開発マシンのメモリにストアする手段、及び該標準入出力情報に関する標準入出力シンボルテーブルと標準入出力割り込みテーブルとを該開発マシンのメモリか該エミュレータのメモリの何れかにストアする手段とをそれぞれ設けるとともに、該被開発装置のマイロプロセッサのプログラムの実行情報と該標

準入出力割り込みテーブルの情報とを比較検出する手段を該エミュレータに設けておき、該エミュレータが該被開発装置のマイロプロセッサのプログラムを制御し実行させるに際し1命令毎に当該プログラムの実行情報をエミュレータにて読みとり、該比較検出手段において当該実行情報と該割り込みテーブルの情報とを比較し、両者が一致した場合にはエミュレータは該被開発装置のマイロプロセッサのプログラムの実行停止を指示すると共に、当該標準入出力に関する関数情報を該標準入出力シンボルテーブルから選択して該開発マシンへ送り、該標準入出力関数情報に該当する標準入出力サブルーチンを該開発マシンに実行させる被開発装置のマイクロプロセッサのプログラム開発システムである。

つまり、本発明の基本的技術思想は開発マシンと被開発装置のマイロプロセッサ及びエミュレータとからなるクロス開発環境下でのマイロプロセッサのプログラム開発において、該開発マシンにおけるコンパイラが出力するオブジェクトファイル内に該被開発装置のマイロプロセッサのネイティブなコードとセルフ開発環境下で一般に用意されている標準入出力機能を実行させるための情報とを含ませ、該情報の出力をエミュレータが識別して該開発マシンの入出力機能に当該標準入出力機能を代行させるように構成されている被開発装置のマイクロプロセッサのプログラム開発及び評価システムである。

ここで、上記の開発マシンとは被開発装置に於けるマイクロプロセッサのプログラムの開発を実行する為にマイクロプロセッサを主体としキーボード、表示装置等を含んで構成されている機器を指すものである。

〔作　用〕

本発明にあっては、開発マシンとマイロプロセッサを有する被開発装置及びエミュレータとからなるクロス開発環境下での被開発装置のマイロプロセッサのプログラム開発システムにおいて、コンパイラが出力する開発マシンにロードされるオ

プジェクトファイル内に該被開発装置のマイロプロセッサのネイティブな制御情報と標準入出力機能を実行させるための標準入出力情報とを含ませておき，該被開発装置のマイロプロセッサのネイティブな制御情報に基きエミュレータを介して該被開発装置のマイロプロセッサのプログラムを実行させると共にその実行情報をエミュレータにおいて該標準入出力情報と比較，識別させ，該両情報が一致した時にエミュレータが該被開発装置のマイロプロセッサのプログラムの実行を停止すると同時に，当該標準入出力に関する情報を開発マシンへ送り該標準入出力機能の実行を該開発マシンの標準入出力機能に代行させるものであるからクロス開発環境下においてもセルフ開発環境下におけると同様にパソコンのＣＲＴを見ながら被開発装置のマイロプロセッサのプログラムにおけるデバッグを実行することが出来る。

〔実施例〕

以下に本発明を実施例の形で詳しく説明する。

まず本発明におけるようなクロス開発環境下において、セルフ開発環境におけると同じ標準入出力関数を用いて被開発装置のマイロプロセッサのプログラムを開発，評価する場合には例えば第２図に示すように０００番地からＦＦＦ番地まで構成されたクロス開発用プログラムが実行されている間にａａａ番地であるデータ("Var")がどう変化しているかを見たい場合"printf "以下の文を挿入し（例えば"printf "("ａ＝%ｄ",Var)として変数 Var の値を"ａ＝1234"の形式で１０進表示するよう指示）これを実行するに当たっては、該被開発装置のマイロプロセッサではこれを表示出来ないため開発マシンのパソコンのマイクロプロセッサを使ってこの情報を画面に出すことにしたものである。従って被開発装置のシステム上のマイクロプロセッサは本来ずっと動いているが"printf"文が挿入されていた箇所に出合うとパソコン上のマイクロプロセッサがそれを検知し，画面表示のため"Var"のデータをエミュレータを通じて見に行きそのデータ

をパソコンの方で例えば"ａ＝1234"というような表示をＣＲＴにさせることになる。

かかるシステムを実行するための基本構成図が第１図に示されており，開発マシン１と温風ヒータ、エアコン等の被開発装置３との間にエミュレータ２を介在させるものであり、開発マシン１には当然オペレーティングシステム４、及びエミュレータ制御ソフト６等が含まれている。又，かかる本発明のシステムにおいて開発マシンに関する基本ソフトを含むコンパイラ５が出力するオブジェクトファイル７内に被開発装置３のマイロプロセッサ１３のネイティブな情報コードとコンパイラのソースプログラムにおいて記述された標準入出力情報，例えばＣ言語であれば、getc，putc，printf，fprintf，scanf，ｆscanf　等の標準入出力関数データをデバッグ情報として含ませるものである。

この場合両者は互いに異なるオブジェクトレコードタイプを用いて含ませることが出来る。

その他，該オブジェクトファイル７にはネイティブ情報のシンボル情報と標準入出力情報のシンボル情報とを含ませておくことも出来る。

その後、本発明においては第１図に示されるようにエミュレータを介して該プログラムを開発評価するに当たり，ローダブルオブジェクトファイル７内に含まれるネイティブ情報，標準入出力情報，ネイティブ情報のシンボル情報及び標準入出力情報のシンボル情報の内，ネイティブコードはエミュレタのメモリ１２にロードし、その他は後述する種々の形式で開発マシンのメモリ８又はエミュレータのメモリ９にロードしておく。

そしてプログラムの実行に当たってはブレークポイントで指定されたシンボルの種別によりエミュレーションメモリの実行か標準入出力の代行かを判断してその実行を切り分ける機能をエミュレータ２に持たせておくものである。第１図において７はローダブルオブジェクトのファイル，７'はクロスコンパイラが出力する絶対形式オブジェクトプログラムのレコード形式である。又８はホストマシンメモリで９及び１２はエミュレータの

メモリでありそれぞれその内にロードされるデータの種類を示している。

尚，該エミュレータメモリ内のデータは開発マシンのメモリ８に含ませておくこともできる。

本発明において使用されるエミュレークは被開発装置のマイロプロセッサのプログラムを制御する機能を有しており該マイロプロセッサとその周辺リソースを必要な時点でエミュレータのハードを用いて代行する機能を有するものであればよい。

通常被開発装置のマイロプロセッサのプログラムはいったん走り出したら被開発装置内で実行されるから好きな番地では止められないためエミュレータが該被開発装置のマイロプロセッサの動作を止めたり進めたりする。具体的には、一般にプローブと称するエミュレータ本体（ハード）から引き出されたリード線の先端に取り付けられているソケットを被開発装置からマイロプロセッサを取り外した跡にさし込んでエミュレータのマイクロプロセッサにより被開発装置のプログラムを制御する（つまりプログラムを走らせる、停める、

どの番地まで実行するか等）ものであってハードそのものはそれほど複雑なものでなくてもよい。

次に本発明にあっては，本システムを作動される為の基本ソフトを含むクロスコンパイラを準備する必要がある。

係るクロスコンパイラは例えば，MS-DOS（マイクロソフト社のオペレーティングシステム®）環境の下で制御用マイクロコントローラのクロスコンパイラを作成する。このクロスコンパイラは printf，scanf等のＣ言語の標準入出力関数がソースプログラムに記述されていると標準入出力レコードを出力する。又、本発明では，該標準入出力レコードから，標準入出力関数に対応するホストマシンコードを開発マシン１のマイクロプロセッサ（ホストコンピュータ）側でサブルーチンの形でメモリ８中にロードしておく。

一方エミュレータ２は該プログラムのロード時に標準入出力レコードに対応した標準入出力割り込みテーブル１１と標準入出力シンボルテーブル１０とを作成しメモリ９にストアしておく。

尚，前記したように，該標準入出力割り込みテーブル１１と標準入出力シンボルテーブル１０とは開発マシンのメモリ８にストアしておくこともできるがデーター処理のスピードが低下するため該テーブルを開発マシン側に設けることは得策ではない。

以下に本発明に於けるシステムの実行手順を第３図のフローチャートに従って説明すると，エミュレータが被開発装置のプログラムの実行中（step a）にその実行アドレスをエミュレータに転送し（step b），該アドレスがエミュレータ（ハード）上のメモリ（ＲＡＭ）に作成された該標準入出力割り込みテーブルの割り込みアドレスと一致するかを一命令毎に比較し（step c）、そのテーブルのアドレスの１つと一致したら例えば現行アドレスと該テーブル中のアドレス 200h（"printf" 文）と一致したらその情報にもとづきエミュレータが被開発装置のプログラム実行を中止する（step d）と共に関数情報を選択し（step e）、これを開発マシンのマイクロプロセッサに送り（step f）標

準入出力関数（プリント或いはＣＲＴ表示等）の処理を割り込み実行させる（step g）ものである。

プログラム実行時に該標準入出力割り込みアドレスをソフトウェアのブレークポイントと考え、割り込みをかけ、その時点でエミュレータは被開発装置のマイクロプロセッサをホールドさせ、標準入出力割り込みテーブルの情報を参照することによりどの標準入出力関数部分を実行するかを判断して制御をホストマシンに移す。ホストマシンは標準入出力処理を実行し終了したら制御をエミュレータに戻すのである。

この処理が終了後エミュレータは被開発装置のシステムの実行を再開する（step h）。

又他の具体例としては、上記標準入出力割り込みテーブルを開発マシンのマイクロプロセッサに作成しておき、被開発装置のシステムの実行中にその実行状態をエミュレータを介して開発マシンのマイクロプロセッサに送りそこで前記と同じ比較処理を行わせ実行アドレスが該テーブル上の割り込みアドレスの１つと一致した場合にはその情報

をエミュレータに戻し上記と同じ処理を行うようにしても良い。本発明では，ホストマシンのマイクロプロセッサのプログラムでエミュレータを制御しつつエミュレータのハードに指示を送りエミュレータに被開発装置のマイクロプロセッサの情報を読み込ませそれを開発マシンに転送させるか，又エミュレータ自身で該情報を読みとり比較してデータがどう変わったかを確認する。

次に本発明において使用されるローダブルオブジェクトファイルのデーター構造及び標準入出力関数、同シンボルテーブル、或いは同割り込みテーブル等のデーター構造について説明する。

第4図に示すようにこのオブジェクトファイルはレコードと呼ばれる形の単位(a)から作成されるのであり、従ってこのファイルはこのデータブロック(a)の集まりから構成されたものである。

以下該データブロックについて簡単な例を用いて説明すると、その構成はレコード識別番号①、レコード長②及びレコードの内容③から出来ている。このレコードには大別してデータレコードと

デバッグ情報レコードとがありその③についての詳細を次に説明するが上記①、と②の部分については省略する。

まずデータレコードの中味は第5図に示すような単位構成(b)となっていて、どこのアドレスからロードさせるかを示すプログラムロードアドレス①、以下に続く③の部分のデータのバイト長を示すデータ長②及びメモリにロードするデータの内容③とから構成されており、①のアドレスにより③のデータをメモリにロードするものである。

このデータレコードをエミュレータが読み込んでエミュレーションメモリにロードするとそのロード結果は第6図に示すようになる。例えば①のアドレスが 200h、データ長②が16バイト、データの内容③が③＝010203…であったとするとこれがエミュレーションメモリ15の斜線の部分にロードされる。

一方デバッグ情報レコードについてみると、デバッグ情報とはエミュレーションを円滑に行うために用意された主にシンボル情報であり、例えば

数値、変数、ジャンプ先の記号からなるもので、コンパイラやアセンブラが出力するものである。

このレコードの単位構成(c)は第7図に示すようにシンボルタイプ①とシンボルデータ②及びシンボル定義番号③から構成されている。

ここでシンボルタイプ①は一般的なものとして数値シンボル、変数、分岐先ラベル（ジャンプ先のラベル）が含まれる。

又シンボルデータ②はシンボルが持っているデータで数値シンボルであれば数値そのもの、変数、か更にはラベルであればそこにアドレスが格納される。

又シンボル定義番号③はシンボルがコンパイラやアセンブラで定義された順に付けられるシリアル番号である。

このレコードをエミュレータが開発マシンのマイクロプロセッサのメモリに又はエミュレータ上のメモリ（RAM）にロードすると第8図に示すシンボルテーブル16が該マイクロプロセッサのメモリ（ホストマシンメモリ）8上に又はエミ

ュレータのメモリ9（RAM）上に形成される。第8図中ラベル1（16－1）が"Var"（変数）とすると例えば Sym1－（1）(シンボルタイプに相当する）を1、Sym1－（2）シンボルデータに相当する）を 200h、Sym1－（3）（シンボル定義番号に相当する）を1のように形成する。ラベルn（16－3）における Symn－（1），Symn－（2），Symn－（3）も同様である。

次に本発明に使用される標準入出力レコードの構成について述べる。高級言語を用いた標準入出力関数である例えば printf，fputc，fgetc，fprintf，fscanf 等は一般にはコンソール入出力関数と考えてよく入力はキーボードによるインプット、出力はテレビ等に表示することを示している。そこで該標準入出力関数がソースプログラム内に記述されるとコンパイラは第9図に示すような標準入出力レコード(d)を出力する。

該レコード（d）の構成は標準入出力関数識別番号を番号付けする部分①、該標準入出力関数が記述されたアドレスを格納する標準入出力関数割

込アドレス部分②と標準入出力の関数情報を格納する部分③とから成っており、前記標準入出力関数識別番号を予め例えば

1 : printf
2 : fputc
3 : fgetc
4 : fprintf
5 : fscanf

のように割りつけておき，このいずれかを①の部分に格納する。又上記標準入出力関数割り込みアドレスには例えば当該関数である " printf " が 200h 番地であるとするとこの部分②には 200h となり、このアドレスの処をプログラムが実行しようとすると割り込みが入り " printf " の処理をするよう命令が出る。更に関数情報の部分③には組み込み関数特有の情報が入るものであり、このフィールドは各関数によって異なる。このデータの構成
は膨大であるのでここでは " printf " を使用する場合を一例として説明する。

今C言語で " printf " の書式は次のように表される。

printf (format, arg1,arg2, … argn)

ここで arg1 … argn は表示したいデータであり、そのデータ構成は第10図に示すようになっている。

同図中①は文字列でformatの内容が入る。即ち、どういう形で表示したらよいかの指示文を入れるものである。例えば printf ("ＡＢ＝%d",Var) としておくとＶａｒの変数の値を１０進で表示せよという指示となり今Ｖａｒの値が１２であればＣＲＴ上にはＡＢ＝１２と表示される。

次にパラメーター②－１，②－２～②－ｎには前記したシンボル情報で定義したシンボル定義番号を入れる。これはそれぞれ arg1,arg2, … argnに対応する。シンボル定義番号が判ればその値が格納されている実際のアドレスがテーブルを参照することで判る。つまり第8図におけるラベル16－１の sym1－（３）の１、或いはラベル16－３の symn－（３）のＮがシンボル定義番号

であり開発マシン又はエミュレータのマイクロプロセッサ上のメモリに作った該シンボルテーブルを参照すればその値は格納されている実アドレスが判る。
このレコードをエミュレータがロードすると第１１図に示すような標準入出力割り込みテーブル１７が開発マシンのマイクロプロセッサのメモリ上もしくはエミュレータのＲＡＭ上にロードする毎に１回作成される。

このテーブルは第１１図のとおり，標準入出力割り込みアドレス①、標準入出力関数識別番号②及び関数情報格納アドレス③から構成されている。

該標準入出力割り込みアドレス①は、例えば前述のデータレコードの２番目に当たるアドレス 200hに相当するもので、つまりプログラムのどこに来たら標準入出力を割り込ませ実行させるかを示す。又標準入出力関数識別番号②としては、例えば " printf " であれば１がここに入る。更に関数情報格納アドレス③には、前記の第９図に示す関数情報が入ることになるが、これを全部

格納することは繁雑であるのでこの情報を別のテーブル④に作っておきそのアドレスのみを③に格納するようにしてもよい。尚かかる標準入出力割り込みテーブル１７は開発マシンのマイクロプロセッサ（所謂パソコン）上に設けてもよいがエミュレータにメモリを増設しておいてそのテーブルを設けることによって処理スピードを上げることが出来る。

この標準入出力割り込みテーブル１７の標準入出力割り込みアドレスがいくつか出来るが、これ等全てをソフトウェアブレークポイントと考え、１命令の実行毎にコンパレート即ち、該ブレークポイントと現行アドレスとが一致するかどうかを調べ、一致した場合にはそれぞれの標準入出力処理の実行をエミュレータが開発マシンのホストコンピュータ（パソコン）に指示する。その処理が終了した場合にはその制御を本来のプログラムに戻すのである。従来このようなプログラムの評価にはリアルタイムで実行しマイクロプロセッサのスピードをいかに落とさないで調べて行くかが重

要な問題となるためハードウェアブレークポイントを通常設けハードウェアでプログラムを比較していくがプログラムのスピードを落とさないためハードウェアブレークポイントはせいぜい4つくらいしか設けられない。一方ソフトウェアブレークポイントによりプログラムを調べる方法では1命令毎にあるかどうかを調べるためエミュレータのスピードが落ちるのでリアルタイム処理には適していない。然しながら本発明においては標準入出力関数を利用して処理することは画面に表示するものであってその間プログラムの実行をストップさせるものであるからリアルタイム処理の問題はなくなり、論理チェックだけであるためソフトウェアブレークポイントを援用しえるのである。

そこで、開発マシンのマイクロプロセッサ又はエミュレータ上のコンパレータではソフトウェアのブレークポイントとしての割り込みアドレスと現行アドレスとを全て比較し、割り込みアドレスと一致したらエミュレータがそれを認識し、どういう標準入出力割り込み関数を使用するかを関数

(31)

情報をみて決定しそれを開発マシンのマイクロプロセッサに指示する。又該マイクロプロセッサには予め用意された例えば標準入出力の処理のためのサブルーチン群を用いていったんその処理結果をＣＲＴに画面表示する。

表示が終了したらエミュレータは本来の被開発装置のシステムの制御にもどって次のプログラムを実行する。エミュレータは被開発装置のマイクロプロセッサのプログラムをみているのでその時の実行されている処理のアドレスは確認出来るからそのアドレス情報を開発マシンのマイクロプロセッサに伝達する。この通信時間はかなりかかるので前記のように当該テーブルをエミュレータに設けることが好ましい。このアドレスが標準入出力割り込みテーブルの各フレームのアドレスと比較し全部比較して一致しなければ次のステップに移るし又一致するのがあれば上記した処理を実行する。

本発明においてはハードブレークポイントを必ずしも設ける必要はないが、リアルタイム的な処

(32)

理をする場合には当然設ける必要がある。

〔発明の効果〕

以上説明したように、本発明によれば、エミュレータを用いたクロス開発環境においても、セルフ開発環境と変わらない高級言語特有のデバッグが可能となり、スタブと呼ばれる printf 文をソースプログラムの途中に多数置くことによるデバッグ手法が使用可能となる。

また、大型汎用のプログラミング環境に慣れた人間でもマイコン制御用のプログラムに異和感なく入ることができる。

4. 図面の簡単な説明

第1図は本発明に係るマイクロプロセッサのプログラム開発システムの基本構成図である。

第2図は本発明に使用されるプログラムの例を示す図である。

第3図は本発明に係るシステムの実行フローチャートである。

第4図は本発明で使用されるオブジェクトファ

(33)

イルのデーター構成を示す図である。

第5図は本発明で使用されるデータレコードの構成を示す図である。

第6図はデータレコードがエミュレータのメモリにロードされた場合のファイル状態を示す図である。

第7図は本発明で使用されるデバッグ情報レコードの構成を示す図である。

第8図は本発明で使用されるシンボルテーブルの例を示す図である。

第9図は本発明で使用される標準入出力レコードの構成を示す図である。

第10図は第9図の標準入出力レコードのうちの関数情報に関するフィールドの構成を示す図である。

第11図は標準入出力割り込みテーブルの例を示す図である。

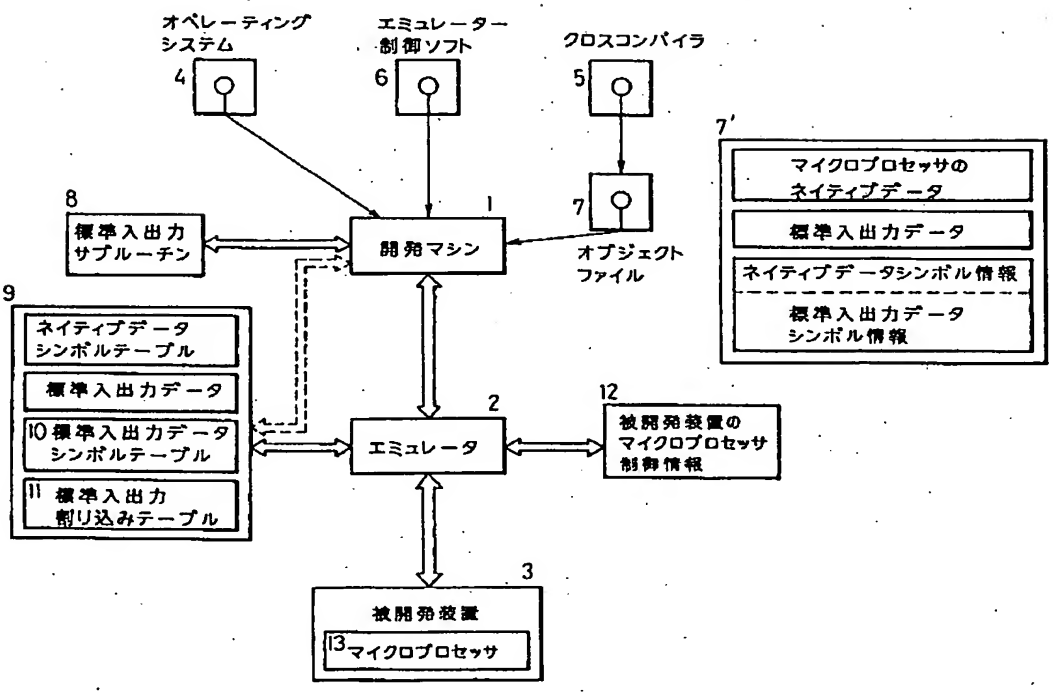
1…開発マシン、　2…エミュレータ、
3…被開発装置
7…クロスコンパイラ

(34)

7′ …クロスコンパイラが出力する絶対形式の
　　オブジェクトプログラムのレコード形式
8 …開発マシンのメモリ
9，12…エミュレータのメモリ
10…標準入出力データシンボルテーブル
11…標準入出力割り込みテーブル
13…マイクロプロセッサ

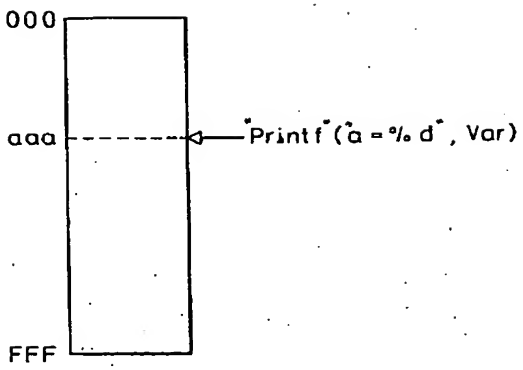
特許出願人
富士通株式会社
富士通マイコンシステムズ株式会社
特許出願代理人
弁理士　宮　木　　朗
弁理士　石　田　　敬
弁理士　平　岩　賢　三
弁理士　山　口　昭　之
弁理士　西　山　雅　也

(35)



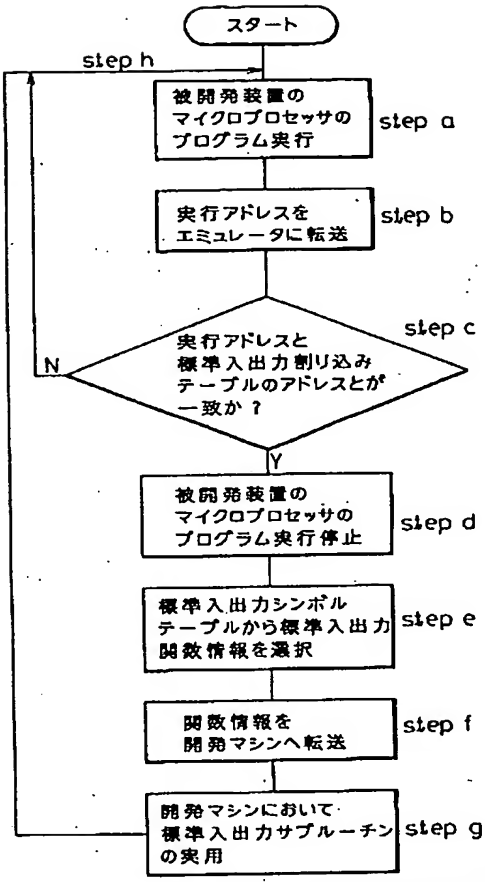
本発明におけるマイクロプロセッサの
プログラム開発の基本構成図

第 1 図



標準入出力情報が挿入されているプログラムの例を示す図

第 2 図



本発明のマイクロプロセッサのプログラム開発の実行フローチャート図

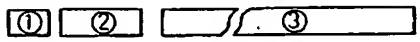
第 3 図

(a) 

① :レコード識別番号
② :レコード長
③ :レコード内容

オブジェクトファイルとのレコード構成図

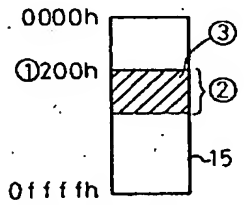
第 4 図

(b) 

① :プログラムロードアドレス
② : データ長 (③のバイト長)
③ : データ

データレコードの構成図

第 5 図



データレコード
①: 200h
②: 16バイト
③: 010203・・・

データレコードがエミュレータに
ロードされた時のファイルの構成図

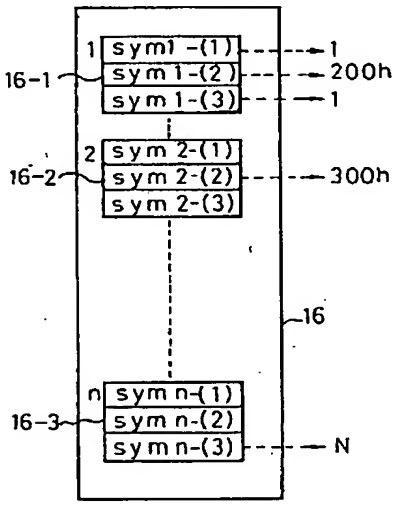
第 6 図

(c) 

① : シンボルタイプ
② : シンボルデータ
③ : シンボル定義番号

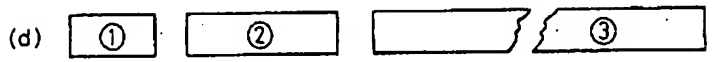
デバッグ情報レコードの構成図

第 7 図



シンボルテーブルの構成図

第 8 図



①：標準入出力関数識別番号

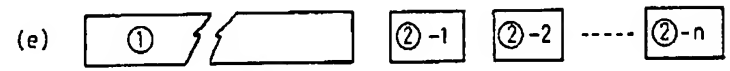
(例) (1)： printf
(2)： fputc
(3)： fgetc
(4)： fprint f
(5)： fscanf

②：標準入出力割り込みアドレス

③：関数情報

標準入出力レコードの構成図

第 9 図

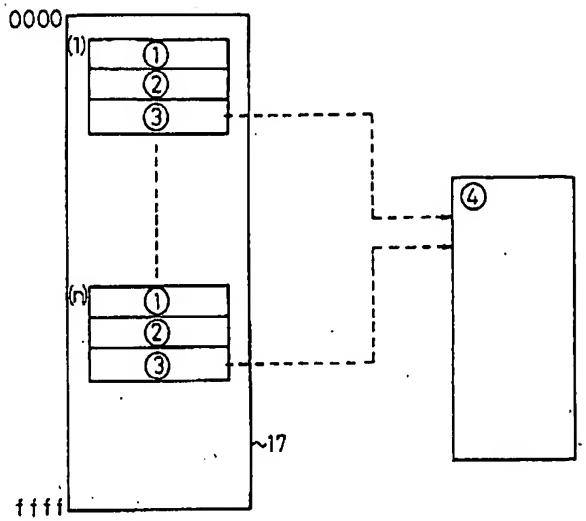


①：format 文字列

②：シンボル定義番号

第9図の関数情報に関するフィールドの構成図

第 10 図



①：標準入出力割り込みアドレス

②：標準入出力関数識別番号

③：関数情報格納アドレス

④：関数情報テーブル

標準入出力割り込みテーブルの構成図

第 11 図

特開平 2—146630(13)